Determinacy of probabilistic ω-languages with strict threshold semantics

Wenjuan Li*[1] and Kazuyuki Tanaka†[1]

[1]Beijing Institute of Mathematical Sciences and Applications, Beijing, China

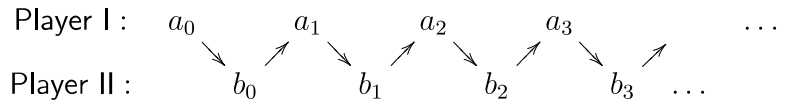
Abstract

Probabilistic automata are a natural generalization of non-deterministic finite automata in the sense of replacing the non-determinism with probabilistic distributions as introduced by Rabin (1963). In this paper, we review several kinds of probabilistic automata on ω-languages and introduce our results of the determinacy strength of some probabilistic ω-languages with threshold semantics.

1 Introduction

We are interested in the determinacy strength of infinite games whose winning sets are recognized by probabilistic automata.

Consider a Gale-Stewart game $\mathbb{G}(X)$, where $X \subseteq A^\omega$ is called a winning set. Player I and player II select an element of A alternatively. A strategy for player I (II) is a mapping $\sigma : \bigcup_{n\in\omega} A^{2n} \to A$



($\tau : \bigcup_{n\in\omega} A^{2n+1} \to A$). Finally, they produce an infinite play x. Player I is said to win with a play x in $\mathbb{G}(X)$ if $x \in X$. Roughly speaking we say σ is a winning strategy for player I if it guarantees that I wins when he follows $\forall n\ a_n = \sigma(a_0 b_0 \cdots a_{n-1} b_{n-1})$, while τ is a winning strategy for player II if it guarantees that II wins when she follows $\forall n b_n = \tau(a_0 b_0 \cdots a_{n-1} b_{n-1})$. A Gale-Stewart game $\mathbb{G}(X)$ is said to be determined if one of the two players has a winning strategy. In this case, we call the winning set X itself determined. Let $\Gamma \subset \mathcal{P}(A^\omega)$, where $\mathcal{P}(A^\omega)$ is the power set of A^ω. By Γ-Det, we denote that "Every set $X \in \Gamma$ is determined".

Dating back to Büchi and Landweber [1], they proved that $\mathbf{REG}_\omega$, the class of ω-regular languages, accepted by non-deterministic Büchi automata, is determined with computable winning strategies.

*liwj@bimsa.cn

†tanaka@bimsa.cn

To increase the expressive power of finite automata, one way is to add stacks, by which the ω-languages are given by pushdown automata. Walukiewicz [17] showed $\mathbf{DCFL}_\omega$, the class of deterministic context-free ω-languages accepted by deterministic Muller pushdown automata, is determined with computable winning strategies. Finkel [8] proved that the deteminacy of $\mathbf{CFL}_\omega$, context-free ω-languages accepted by pushdown automata with a Büchi acceptance conditions, is not provable in the set theory ZFC. For the determiancy of pushdown ω-languages with weaker acceptance condition, e.g., safety, reachability, Li and Tanaka [9] investigated the determinacy of such games in the framework of reverse mathematics. Their results also hold for the ω-languages accepted by the corresponding non-deterministic/deterministic 2-visibly pushdown automata and 1-counter automata.

Another generalization of finite automata is to replace the non-deterministic transition relation with a probability distribution, namely probability automata. Rabin [15] defined probabilistic automata on finite words. In this paper, we review the studies on ω-languages accepted by probabilistic automata and introduce our studies on the determinacy strength of infinite games for such languages.

2 From non-deterministic to probabilistic

First, we brief review how a finite automaton accepts ω-languages.

Example 2.1. *A finite automaton with an input alphabet $A = \{a, b\}$, set of states $Q = \{q_0, q_1\}$ is illustrated as follows, where q_0 is both an initial and final state (indicated with double circle):*

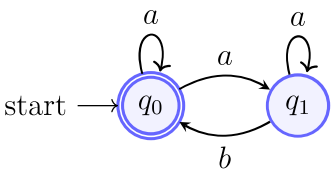


Figure 1: A non-deterministic Büchi automaton $\mathcal{M}$

A run on input word is an infinite sequence of Q. For instance, a run on the input $(ab)^\omega = ababab\cdots$ is as follows:

$$q_0 \xrightarrow{a} q_1 \xrightarrow{b} q_0 \cdots \xrightarrow{a} q_1 \xrightarrow{b} q_0 \cdots$$

An input word is accepted *by a Büchi automaton if there exists a run visiting a state in $F(= \{q_0\})$ infinitely many times.*

The infinite word $(ab)^\omega$ is accepted by $\mathcal{M}$. There might be more than one run on a single input due to the non-determinism of the automaton. The ω-language accepted by $\mathcal{M}$ is $\mathcal{L}(\mathcal{M}) = (ab \cup a)^\omega$. Note that player I has a winning strategy in the game $\mathbb{G}(\mathcal{L}(\mathcal{M}))$ by always playing a. As we mentioned in the Section 1, every ω-language accepted by finite automata is determined [1].

As shown in Figure 1, $\mathcal{M}$ is nondeterministic in the sense that at state q_0, by reading a, it can simultaneously stay in state q_0 and move to q_1. It is natural to consider a replacement of the non-determinism with a probability distribution as in Figure 2.

Example 2.2 (cf. [5]). *The probabilistic Büchi automaton $\mathcal{P}$ in Figure 2 accepts the language with a positive semantics*

$$\mathcal{L}^{>0}(\mathcal{P}) = \left\{ a^{k_1} b a^{k_2} b a^{k_3} b \cdots : k_1, k_2, k_3 \ldots \mathbb{N}_{\geq 1} \text{ such that } \prod_{i=1}^{\infty} \left(1 - \frac{1}{2^{k_i}}\right) > 0 \right\}.$$

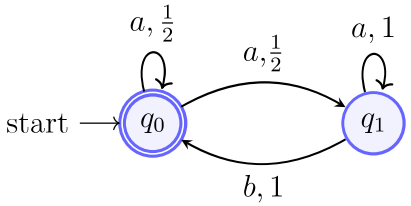


Figure 2: A probabilistic Büchi automaton $\mathcal{P}$

We can see that player II has a winning strategy in the game $\mathbb{G}(\mathcal{L}^{>0}(\mathcal{P}))$, by always playing b after every a selected by player I, producing an infinite play $abababab\cdots \notin \mathcal{L}^{>0}(\mathcal{P})$.

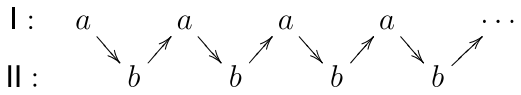


Figure 3: A winning play of player II in $\mathbb{G}(\mathcal{L}^{>0}(\mathcal{P}))$

Now we give the formal definition of probabilistic automata.

Definition 2.1. *A probabilistic automaton is a tuple $\mathcal{P} = (Q, A, \delta, \delta_0, F, \mathsf{Acc})$, where*

- *Q is a finite set of states,*
- *A is a finite input alphabet,*
- *transition probabilistic distribution $\delta : Q \times A \times Q \to [0,1]$ such that for all $q \in Q, a \in A$, $\sum_{q' \in Q} \delta(q, a, q') \in \{0, 1\}$,*
- *$\delta_0 : Q \to \{0, 1\}$ is the initial distribution such that $\Sigma_{q \in Q} \delta_0(q) = 1$,*
- *F is the set of accepting states, and*
- *Acc is the acceptance condition, e.g., reachability, Büchi.*

The probabilistic automaton $\mathcal{P} = (Q, A, \delta, q_0, F, \mathsf{Acc})$ as shown in 2.2 is equipped with (probability) transitions, $\delta_0(q_0) = 1$, $\delta(q_0, a, q_0) = \frac{1}{2}$, $\delta(q_0, a, q_1) = \frac{1}{2}$, $\delta(q_1, a, q_1) = 1$, and $\delta(q_1, b, q_0) = 1$.

Similar as the finite automata, a run of a probabilistic automaton is a sequence of states produced by valid transitions and the acceptance of a run is determined by the acceptance condition, for instance Büchi condition. The acceptance condition Acc we considered here include:

- Safety (or Π_1) acceptance condition

$$\text{there is a run } (q_i)_{i \geq 0} \text{ of } \mathcal{P} \text{ on } \alpha \text{ s.t. } \underbrace{\forall i, q_i \in F}_{\texttt{Safety}(F)}$$

- Reachability (or Σ_1) acceptance condition

$$\text{there is a run } (q_i)_{i\geq 0} \text{ of } \mathcal{P} \text{ on } \alpha \text{ s.t. } \underbrace{\exists i, q_i \in F}_{\texttt{Reach}(F)}$$

- $(\Sigma_1 \wedge \Pi_1)$ acceptance condition with F_r, $F_s \subset Q$

$$\text{there is a run } (q_i)_{i\geq 0} \text{ of } \mathcal{P} \text{ on } \alpha \text{ s.t. } \quad \texttt{Reach}(F_r) \wedge \texttt{Safety}(F_s)$$

- Büchi (or Π_2) condition

$$\text{there is a run } (q_i)_{i\geq 0} \text{ of } \mathcal{P} \text{ on } \alpha \text{ s.t. } \underbrace{\forall i \exists j > i, q_j \in F}_{\texttt{Büchi}(F)}$$

- Co-Büchi (or Σ_2) acceptance condition

$$\text{there is a run } (q_i)_{i\geq 0} \text{ of } \mathcal{P} \text{ on } \alpha \text{ s.t. } \underbrace{\exists i \forall j > i, q_j \in F}_{\texttt{co-Büchi}(F)}$$

- Δ_2 acceptance condition with F_b, $F_c \subset Q$:

  there is a run r on α satisfying $\texttt{Büchi}(F_b)$ and there is a run r on α satisfying $\texttt{co-Büchi}(F_c)$

Given $\mathcal{P}$ and an input α, let $\mathsf{Acc}(F, \alpha)$ denote the set of runs of α satisfying acceptance condition Acc, that is, $\mathsf{Acc}(F, \alpha) = \{\rho \in Q^\omega \mid \rho \text{ satisfies } \mathsf{Acc}\}$. The acceptance of a word α is evaluated by the accepting measure $\mu^{\mathsf{Acc}}_{\mathcal{P},\alpha}$ on the set $\mathsf{Acc}(F, \alpha)$ under a certain **semantics**. Table 1 lists several kinds of semantics and their corresponding ω-languages.

Table 1: ω-languages defined by different semantics

semantics	measure	ω-languages
positive	$\mu^{\mathsf{Acc}}_{\mathcal{P},\alpha} > 0$	$\mathcal{L}^{>0}(\mathcal{P}) = \{\alpha \in A^\omega \mid \mu^{\mathsf{Acc}}_{\mathcal{P},\alpha} > 0\}$
almost-sure	$\mu^{\mathsf{Acc}}_{\mathcal{P},\alpha} = 1$	$\mathcal{L}^{=1}(\mathcal{P}) = \{\alpha \in A^\omega \mid \mu_{\mathcal{P},\alpha} = 1\}$
strict threshold	$\mu^{\mathsf{Acc}}_{\mathcal{P},\alpha} > \lambda$	$\mathcal{L}^{>\lambda}(\mathcal{P}) = \{\alpha \in A^\omega \mid \mu^{\mathsf{Acc}}_{\mathcal{P},\alpha} > \lambda\}$, for $\lambda \in (0,1)$
threshold	$\mu^{\mathsf{Acc}}_{\mathcal{P},\alpha} \geq \lambda$	$\mathcal{L}^{\geq\lambda}(\mathcal{P}) = \{\alpha \in A^\omega \mid \mu^{\mathsf{Acc}}_{\mathcal{P},\alpha} \geq \lambda\}$, for $\lambda \in (0,1)$

We define the class of ω-languages accepted by probabilistic automata with acceptance Acc and strict threshold semantics as follows

$$\mathbb{L}(\mathbf{PA}^{>\lambda}_{\mathsf{Acc}}) = \{L \subset A^\omega \mid \text{there exists a probabilistic automaton } \mathcal{P} \text{ s.t. } L = \mathcal{L}^{>\lambda}(\mathcal{P})\}.$$

Probabilistic ω-languages of other acceptance condition and semantics can also be defined similarly.

Focusing on the regular property of probabilistic ω-languages, some variants of probability automata are introduced in literature, such as hierarchical probabilistic automata, simple probabilistic automata.

Hierarchical probabilistic automata

Given a natural number k, a probabilistic automaton $\mathcal{H}$ is called **k-hierarchical** if there is a ranking function $\mathrm{rk} : Q \to \{0, 1, \cdots, k\}$ with $Q = \bigsqcup_{j=0}^{k} Q_j = \bigsqcup_{j=0}^{k} \{q \in Q \mid \mathrm{rk}(Q) = j\}$ such that for any $q \in Q$, $a \in \Sigma$, if $i = \mathrm{rk}(q)$, then $\mathrm{Next}(q, a) \subseteq \bigcup_{i \leq \ell \leq k} Q_\ell$ and $\mid \mathrm{Next}(q, a) \cap Q_i \mid \leq 1$. That is, from any state of a certain level, reading any input symbol, the automaton can go to at most one state of the same level and all other branches go to higher levels. We can define $\mathbb{L}(\mathbf{HPA}^{*}_{\mathsf{Acc}})$ for some probability semantics $*$ and acceptance condition Acc.

Simple probabilistic automata

As we will introduce in Section 3, the ω-languages accepted by hierarchical Büchi probabilistic automata with non-extreme semantics (i.e., strict threshold and threshold) lose the regularity property. In [6], a special class of such automata is defined, which is called **simple probabilistic automata** in the sense that it is 1-level hierarchical probabilistic automata and all accepting states belong to level 0. Analogously, its languages are denoted as $\mathbb{L}(\mathbf{SPA}^{*}_{\mathsf{Acc}})$.

Finite probabilistic monitors and robust probabilistic automata

Finite probabilistic monitors are a special kind of probabilistic Büchi automata with an absorbing reject state. They can be regarded as randomized run-time monitoring algorithms or models of open, probabilistic reactive systems with failures captured by a rejecting state. Motivated by application questions, probabilistic monitors with various acceptance conditions, including strong/weak acceptance and strict/non-strict cut points, are defined in [2]. Given a probabilistic monitor $\mathcal{M}$ on A and $\lambda \in (0, 1)$, we can define

$$\mathcal{L}^{<\lambda}_{\mathsf{rej}}(\mathcal{M}) = \{\alpha \in A^{\omega} \mid \mu^{\mathsf{rej}}_{\mathcal{M},\alpha} < \lambda\},$$

and $\mathcal{L}^{\leq\lambda}_{\mathsf{rej}}(\mathcal{M})$, $\mathcal{L}^{\leq 0}_{\mathsf{rej}}(\mathcal{M})$, $\mathcal{L}^{\leq 0}_{\mathsf{rej}}(\mathcal{M})$ can be defined in the same manner. Similarly, we have the classes of ω-languages denoted as $\mathbb{L}((\mathbf{PM}^{*}))$. A probabilistic monitor $\mathcal{M}$ with alphabet A is called x-robust for $x \in (0, 1)$ if there exists ϵ such that for any input $\alpha \in A^{\omega}$, $\mid \mu^{\mathsf{rej}}_{\mathcal{M},\alpha} - x \mid > \epsilon$. We denote the class of such language as Robust.

It is interesting to see that for x-robust probabilistic monitor $\mathcal{B}$ on alphabet A, if $\mathcal{L}^{>\lambda}(\mathcal{B})$ or $A^{\omega} - \mathcal{L}^{>\lambda}(\mathcal{B})$ is Π^{0}_{1}, then it is indeed ω-regular. Intuitively, the ω-regularity of some Π^{0}_{1} languages is equivalent to the regularity of its finite prefixes. Note that notion of robustness generalize the isolated cut-point defined in the context of finite words, and such languages of finite words are regular [15].

Ambiguity of probabilistic automata

By considering the number of different accepting runs on an input, another kind of probabilistic ω-languages that characterize the ω-regular languages can be obtained. An automaton is called **k-ambiguous** if there are at most k different runs on each input for some $k \in \mathbb{N}$. The class of ω-languages accepted by probabilistic k-ambiguous automata is denoted as $\mathbb{L}(k\text{-}\mathbf{PA}^{*}_{\mathsf{Acc}})$. Similarly, **polynomially ambiguous** and **countably ambiguous** automata are defined in [10], whose languages are denoted as $\mathbb{L}(n^{k}\text{-}\mathbf{PA}^{*}_{\mathsf{Acc}})$ and $\mathbb{L}(\aleph_0\text{-}\mathbf{PA}^{*}_{\mathsf{Acc}})$.

3 Topological complexity of probabilistic ω-languages

We start with finite words accepted by probabilistic automata. The cardinality of the class of languages of finite words accepted by probabilistic automata over real numbers equals to cadinality of continuum [15, 14]. Then there must be some languages accepted by probabilistic automata but not accepted by any deterministic finite machine, including deterministic finite Turing machines. However, this is not the case when we consider restricted probabilistic distributions and thresholds as shown below.

Theorem 3.1 ([7]). *If the probabilistic automata is computable (i.e., their transition distributions and the threshold values are computable reals), then the probabilistic languages of finite words accepted by such automata are computably enumerable.*

Now we discuss ω-languages accepted by probabilistic automata and Turing machines. We follows the definition of Turing machines in [16], in which the machines are not required to finish reading the whole tape. Let $\mathbb{L}(\mathbf{TM}_{\mathsf{Acc}})$ ($\mathbb{L}(\mathbf{DTM}_{\mathsf{Acc}})$) denote the class of languages accepted by (deterministic) Turing machines with acceptance condition $\mathsf{Acc} = \Pi_1, \Sigma_1, \Pi_2, \Sigma_2$.

Theorem 3.2 (cf. [16]).

$$\begin{array}{lcll} \mathbb{L}(\mathbf{DTM}_{\Pi_1}) & = & \mathbb{L}(\mathbf{TM}_{\Pi_1}) & = & \Pi_1^0 \\ \mathbb{L}(\mathbf{DTM}_{\Sigma_1}) & = & \mathbb{L}(\mathbf{TM}_{\Sigma_1}) & = & \Sigma_1^0 \\ \mathbb{L}(\mathbf{DTM}_{\Sigma_2}) & = & \mathbb{L}(\mathbf{TM}_{\Sigma_2}) & = & \Sigma_2^0 \\ \mathbb{L}(\mathbf{DTM}_{\Pi_2}) & = & \Pi_2^0 & & \\ \mathbb{L}(\mathbf{TM}_{\Pi_2}) & = & \Sigma_1^1 & & \end{array}$$

By Theorem 3.1 and 3.2, we can obtain

Theorem 3.3. *Assume the probabilistic automata are computable. For $*$ denoting positive, almost-sure, threshold semantics, and $\mathsf{Acc} = \Pi_1, \Sigma_1, \Sigma_2$*

$$\mathbb{L}(\mathbf{PA}^*_{\mathsf{Acc}}) \subseteq \mathbb{L}(\mathbf{TM}_{\mathsf{Acc}}).$$

and in particular for $\mathsf{Acc} = \Pi_2$

$$\mathbb{L}(\mathbf{PA}^*_{\Pi_2}) \subseteq \mathcal{B}(\Sigma_2^0).$$

For threshold semantics, we have the following nice property. Although the original proof can be found elsewhere, e.g. [3], we here present the proof for a straightforward understanding of such a property.

Lemma 3.4. *(1) $\mathbb{L}(\mathbf{PA}^{>\lambda}_{\Pi_2}) = \mathbb{L}(\mathbf{PA}^{>\eta}_{\Pi_2})$ for any $\lambda, \eta \in (0,1)$.*
(2) $\mathbb{L}(\mathbf{PA}^{\geq\lambda}_{\Pi_2}) = \mathbb{L}(\mathbf{PA}^{\geq\eta}_{\Pi_2})$ for any $\lambda, \eta \in (0,1)$.

Proof. Given a probabilistic Büchi automaton $\mathcal{P} = (\{q_0, \cdots, q_{n-1}\}, A, \delta_0, \delta, F)$ with threshold λ, we want to construct a Büchi $\mathcal{P}'$ with threshold η such that $L(\mathcal{P}) = L(\mathcal{P}')$.

Assume $\lambda > \eta$. Suppose $\mathcal{P}' = (\{q_0, \cdots, q_{n-1}, q_n\}, A, \delta'_0, \delta', F)$, where $\mathcal{P}'$ includes a copy of $\mathcal{P}$ plus an extra state q_n which is a sinking reject state and

- $\delta'_0(q_i) = \frac{\eta}{\lambda}\delta_0(q_i)$ for $0 \leq i \leq n-1$

- $\delta'_0(q_n) = 1 - \frac{\eta}{\lambda}$

For any infinite word α, the accepting probability of $\mathcal{P}'$ is $\frac{\eta}{\lambda}\mathsf{Pr}^{>\lambda}_{\mathcal{P}}(\alpha) > \eta$, and thus $L(\mathcal{P}) = L(\mathcal{P}')$.

Assume $\lambda < \eta$. Suppose $\mathcal{P}' = (\{q_0, \cdots, q_{n-1}, q_n\}, A, \delta'_0, \delta', F \cup \{q_n\})$, where $\mathcal{P}'$ consists of a copy of $\mathcal{P}$ plus an extra state q_n which is a sinking accepting state and

- $\delta'_0(q_i) = \frac{1-\eta}{1-\lambda}\delta_0(q_i)$ for $0 \leq i \leq n-1$
- $\delta'_0(q_n) = 1 - \frac{1-\eta}{1-\lambda} = \frac{\eta-\lambda}{1-\lambda}$, since δ_0 is the initial distribution and $\sum_{i=0}^{n} \delta_0(q_i) = 1$

For any infinite word α, the accepting probability of $\mathcal{P}'$ is $\mathsf{Pr}^{>\eta}_{\mathcal{P}'}(\alpha) = \frac{\eta-\lambda}{1-\lambda} + \frac{1-\eta}{1-\lambda}\mathsf{Pr}^{>\lambda}_{\mathcal{P}}(\alpha) > \eta$, and thus $L(\mathcal{P}) = L(\mathcal{P}')$. □

Notice that this property of threshold semantics also holds for other acceptance conditions.

For the relation among ω-languages accepts by variants of probabilistic automata, we summarize as follows, especially for Π_2 (Büchi) condition,

(1) $\mathbf{REG}_\omega \cap \Pi^0_1 = \mathsf{Robust} = \mathbb{L}(\mathbf{PM}^{\leq 0}) \subset \mathbb{L}(\mathbf{PM}^{\leq \lambda}) \subset \Pi^0_1$

$\subset \mathbf{REG}_\omega \cap \Pi^0_2 \subset \mathbb{L}(\mathbf{PM}^{<1}) \subset \mathbb{L}(\mathbf{PM}^{<\lambda}) \subset \Pi^0_2$, cf. [2]

(2) $\mathbf{REG}_\omega \cap \mathsf{Deterministic} = \mathbb{L}(\mathbf{SPA}^{>\lambda}_{\Pi_2}) = \mathbb{L}(\mathbf{SPA}^{\geq\lambda}_{\Pi_2})$

$= \mathbb{L}(\mathbf{HPA}^{=1}_{\Pi_2}) \subsetneq \mathbb{L}(\mathbf{PA}^{=1}_{\Pi_2}) \subsetneq \mathbb{L}(\mathbf{PA}^{\geq\lambda}_{\Pi_2})$, cf. [3, 5]

$= \mathbb{L}(k\text{-}\mathbf{PA}^{=1}_{\Pi_2}) = \mathbb{L}(2^k\text{-}\mathbf{PA}^{=1}_{\Pi_2}) \subset \mathbb{L}(\aleph_0\text{-}\mathbf{PA}^{=1}_{\Pi_2})$, cf. [10]

(3) $\mathbf{REG}_\omega = \mathbb{L}(\mathbf{HPA}^{>0}_{\Pi_2}) \subsetneq \mathbb{L}(\mathbf{PA}^{>0}_{\Pi_2}) \subsetneq \mathbb{L}(\mathbf{PA}^{>\lambda}_{\Pi_2})$, cf. [3, 5]

$= \mathbb{L}(k\text{-}\mathbf{PA}^{>0}_{\Pi_2}) = \mathbb{L}(\aleph_0\text{-}\mathbf{PA}^{>0}_{\Pi_2})$, cf. [10]

$= \mathbb{L}(k\text{-}\mathbf{PA}^{>\lambda}_{\Pi_2}) \subset \mathbb{L}(n^k\text{-}\mathbf{PA}^{>\lambda}_{\Pi_2})$, cf. [10]

(4) $\mathbb{L}(\mathbf{PA}^{=1}_{\Pi_2}) \subset \Pi^0_2$

$\mathbb{L}(\mathbf{PA}^{>0}_{\Pi_2}) = \mathcal{B}(\mathbb{L}(\mathbf{PA}^{=1}_{\Pi_2})) \subset \mathcal{B}(\Sigma^0_2)$, cf. [5]

$\mathbb{L}(\mathbf{PA}^{>\lambda}_{\Pi_2}) \subset \mathcal{B}(\Sigma^0_2)$, $\mathbb{L}(\mathbf{PA}^{\geq\lambda}_{\Pi_2}) \subset \mathcal{B}(\Sigma^0_2)$

(5) $\mathbb{L}(\mathbf{PA}^{>\lambda}_{\Pi_2})$ and $\mathbb{L}(\mathbf{PA}^{\geq\lambda}_{\Pi_2})$ are not comparable, $\mathbb{L}(\mathbf{HPA}^{>\lambda}_{\Pi_2})$ and $\mathbb{L}(\mathbf{HPA}^{\geq\lambda}_{\Pi_2})$ are not comparable, $\mathbb{L}(\mathbf{PM}^{\leq\lambda})$ and $\mathbb{L}(\mathbf{PM}^{<\lambda})$ are not comparable, cf. [6], [2]

where $\mathcal{B}(X)$ is the Boolean combination of X, $\mathbf{REG}_\omega \cap \mathsf{Deterministic}$ denotes exactly the class of ω-languages accepted by deterministic Büchi automata.

4 Determinacy of probabilistic ω-languages of strict threshold semantics

In the following, we will treat probabilistic ω-languages of strict threshold semantics. Such class of languages itself does not depend on the value of the threshold as shown in Lemma 3.4. For simplicity, we set threshold as $\frac{1}{2}$.

Theorem 4.1. *There is an infinite game in* $\mathbb{L}(\mathbf{PA}^{>\frac{1}{2}}_{\Sigma_1\wedge\Pi_1})$ *with only* Σ^0_1*-hard winning strategies.*

The proof is an improvement of our results in [9] and for the self-contained purpose, we re-state some necessary part. We will construct a game associated with a universal two-counter machine $\mathcal{R}$ such that the halting problem of $\mathcal{R}$ is computable in any winning strategy of player II, while player I has no winning strategy.

Each counter of the (universal nondeterministic) 2-counter automaton stores a nonnegative integer. The input is a natural number m, initially stored in one of the counters. Then by the current state and the tests results on whether each counter is zero or not, the automaton goes to the next state and do operations on the two counters by increasing the counter(s) by 1, or decreasing the counter(s) by 1 if the counter is not zero.

A 2-counter automaton is a tuple $\mathcal{R} = (Q, \delta, q_{\mathsf{in}}, F)$, where Q is a finite set of states, $\delta \subseteq Q \times \{0,1\}^2 \times Q \times \{-1,0,1\}^2$ a transition relation, q_{in} an initial state and F a set of halting states. A configuration of 2-counter automata is (q, m, n), where $q \in Q$, and m, n are nonnegative integers in the two counters. For any input natural number m of $\mathcal{R}$, the initial configuration is $(q_{\mathsf{in}}, m, 0)$.

We code a configuration (q, m, n) of a 2-counter automaton as $qa^m b^n$. A *run* for a natural number m on a 2-counter automaton $\mathcal{R}$ is a sequence of configurations such that $q_{\mathsf{in}}a^{m_0}b^{n_0} \mapsto_{\mathcal{R}} q_1 a^{m_1} b^{n_1} \mapsto_{\mathcal{R}} q_2 a^{m_2} b^{n_2} \mapsto_{\mathcal{R}} \cdots$, where q_{in} is the initial state, $m_0 = m$, $n_0 = 0$ and $\mapsto_{\mathcal{R}}$ is defined by the transition relation δ of $\mathcal{R}$. A run is halting if it reaches a halting configuration.

We define a number $m \in L(\mathcal{R})$ if and only if there exists a run on m such that $q_{\mathsf{in}}a^m b^{n_0} \mapsto_{\mathcal{R}} q_1 a^{m_1} b^{n_1} \mapsto_{\mathcal{R}} \cdots \mapsto_{\mathcal{R}} q_s a^{m_s} b^{n_s}$, where $n_0 = 0$ and $q_s \in F$. It is known that a 2-counter automaton, even a deterministic one, is equivalent to a Turing machine [11, 12]. Thus the halting problem for a certain (universal deterministic) 2-counter automaton is Σ^0_1-complete. In the following, by 2-counter automata, we mean deterministic 2-counter automata unless stated otherwise.

Proof. We construct a game where player I (male) can choose any natural number m and ask whether such a number can be accepted by the universal 2-counter automaton $\mathcal{R}$ while player II (female) can choose the $\forall$ or $\exists$ role such that she always wins but player I has no winning strategy. We consider the case where player II plays an $\exists$ role: she challenges to find player I's error after she answers no as shown below.

Player I:	$m \in L(\mathcal{R})$?	$q_0 a^m b^{n_0} \rhd \cdots \rhd q_i a^{m_i} b^{n_i} \rhd q_{i+1} a^{m_{i+1}} b^{n_{i+1}}$
Player II:	yes or no	

In order to win, player II needs to find the error in the sequence of configurations of a halting run of $\mathcal{R}$ with input m provided by player I.

That is, player II checks whether player I has obeyed the following rules.

($r0$) player I has no intension to produce infinite many "1"s in the first round,

($r1$) the sequence of configurations provided by player I is a sequence in the form of $qa^m b^n$ and connected by $\rhd$ (for simplicity $\rhd$ is the code of $\mapsto_{\mathcal{R}}$),

($r2$) the sequence starts with the initial configuration,

($r3$) any two consecutive configurations constitute a valid transition of $\mathcal{R}$, and

($r4$) the sequence of configurations is ended with a halting configuration.

The rules (0), (1), (2), and (4) are easy to check with Σ_1 conditions by finite memory (namely, player I loses with Π_1). For (3), it requires to guarantee the correctness of every pair of successive computations.

The key of the proof for Theorem 4.1 lies in using probabilistic automata to mimic the computation process of $\mathcal{R}$, and in particularly constructing a probabilistic automaton accepting ω-language in the form of $L = \{a^n b^n \sharp \alpha : n \in \mathbb{N}, \alpha \in \{a, b\}^\omega\}$ as in the following example.

Example 4.2. *A weak equality test (Freivalds [1981], Condon and lipton [1989]) is introduced to show how probability automaton accepts $L = \{a^n b^n \sharp \alpha : n \in \mathbb{N}, \alpha \in \{a, b\}^\omega\}$, which provides a probabilistic way using finite bounded memory to check whether two input sequences are of the same length.*

When we treat a^i, we conduct the following three experiments:

X_a^{2i} $=$*True if $2i$ coins are all heads.*
X_a^{i} $=$*True if i coins are all heads.*
X_a^{i+} $=$*True if another i coins are all heads.*

and similarly for b^j, we have X_b^{2j}, X_b^j and X_b^{j+}.

Let $\mathbb{A} = X_a^{2i} \vee X_b^{2j}$, and $\mathbb{B} = \left(X_a^i \wedge X_b^j\right) \vee \left(X_a^{i+} \wedge X_b^{j+}\right)$. As in [6], we define the following events:

(1) $\mathcal{E}_1$ if $\mathbb{B} \wedge \neg\mathbb{A}$,
(2) $\mathcal{E}_0$ if $\mathbb{A} \wedge \neg\mathbb{B}$,
(3) $\mathcal{E}_{\textbf{AllHeads}}$ if $X_a^{2i} \wedge X_a^i \wedge X_a^{i+} \wedge X_b^{2j} \wedge X_b^j \wedge X_b^{j+}$, and
(4) $\mathcal{E}_{\textbf{others}}$ if none of the above cases hold.

Proposition 4.3. *The following hold.*

- $\mathsf{Pr}(\mathcal{E}_1) \geq \mathsf{Pr}(\mathcal{E}_{\textbf{\textit{AllHeads}}})$.
- *If $i = j$, then $\mathsf{Pr}(\mathcal{E}_0) = \mathsf{Pr}(\mathcal{E}_1)$. Note that this value may be less than $\frac{1}{2}$.*
- *If $i \neq j$, $\mathsf{Pr}(\mathcal{E}_0) - \mathsf{Pr}(\mathcal{E}_1) > 3\mathsf{Pr}(\mathcal{E}_{\textbf{\textit{AllHeads}}})$.*

The remaining part is to construct a probabilistic automaton $\mathcal{B}$ such that $\mathcal{L}^{>\frac{1}{2}}_{\Sigma_1 \wedge \Pi_1}(\mathcal{B}) = L$ via imitating the weak equality test. To carry out the test, the automaton needs to memorize the following:

(1) when reading a, it needs to record the value of X_a^{2i}, X_a^i and X_a^{i+} (3 bits)
(2) when reading b, it needs to record not only the value of X_a^{2i}, X_a^i, X_a^{i+} but also X_b^{2i}, X_b^i, X_b^{i+} (6 bits)

which requires $2^3 + 2^6$ *states in total.*

The initial value of the above flag variables are set as True*, which means that no tails have been seen. When some tail appears in the experiment in the scope of some flag variable, say* X_a^{2i}*,* X_a^{2i} *will change to* False *and never change to* True*. Note that the value of* X_a^{2i}*,* X_a^i*,* X_a^{i+}*,* X_b^{2i}*,* X_b^i*,* X_b^{i+} *can only be sequentially changed. Such a process can be captured by finite states with a hierarchical structure. The automaton can be roughly illustrated in Figure 4.*

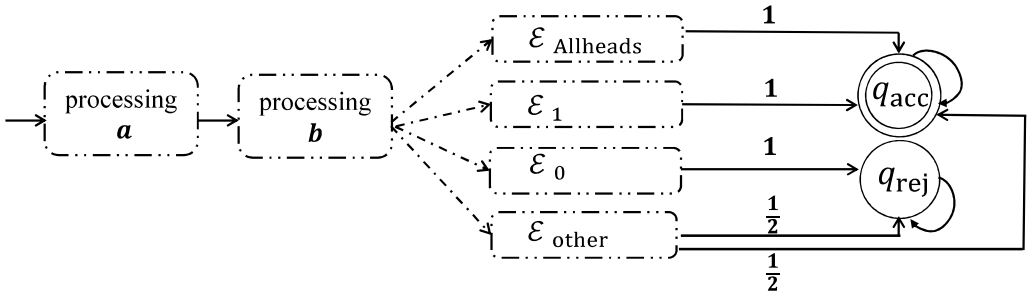


Figure 4: A schematic diagram of $\mathcal{B}$

- *If the weak equality test outputs* $\mathcal{E}_1$*,* $\mathcal{B}$ *goes to* q_{acc} *with probability 1.*
- *If the weak equality test outputs* $\mathcal{E}_0$*,* $\mathcal{B}$ *goes to* q_{rej} *with probability 1.*
- *If the weak equality test outputs* $\mathcal{E}_{\boldsymbol{AllHeads}}$*,* $\mathcal{B}$ *goes to* q_{acc} *with probability 1.*
- *In all other cases,* $\mathcal{B}$ *goes to* q_{acc} *and* q_{rej} *with probability* $\frac{1}{2}$*.*

Then the accepting probability is $\mathsf{Pr}(\mathcal{E}_{\text{Allheads}}) + \mathsf{Pr}(\mathcal{E}_1) + \frac{1}{2}\mathsf{Pr}(\mathcal{E}_{\text{others}})$ *and the rejecting probability* $\mathsf{Pr}(\mathcal{E}_0) + \frac{1}{2}\mathsf{Pr}(\mathcal{E}_{\text{other}})$*. By Proposition 4.3, if* $i = j$*, the accepting probability* $> \frac{1}{2}$*.*

Back to the proof, the winning plays can be accepted by a variant of the above probabilistic automaton as we explain below. To treat $(r1)$, essentially, the probabilistic automaton can accept the sequence including segment like: $q_i a^{m_i} b^{n_i} \rhd q_{i+1} a^{m_{i+1}} b^{n_{i+1}}$ such that

- $m_i = m_{i+1}$ or $m_i = m_{i+1} + 1$ or $m_i = m_{i+1} - 1$, and
- $n_i = n_{i+1}$ or $n_i = n_{i+1} + 1$ or $n_i = n_{i+1} - 1$

The difference from the above example is that we compare the length of two appearances of "a" and also compare the length of two appearances of "b". More specifically, the automaton compare both the length of two "a"-segments and two "b"-segments while scanning from left to right once, and the two "a"-segments ("b"-segments) are not consecutive. To this end, the desired probabilistic automaton just needs a finite memory to store and update the corresponding flags variables.

Now assume that player II has a winning strategy τ, then the halting set for $\mathcal{R}$ is

$$L(\mathcal{R}) = \{m : \text{ player II follows } \tau \text{ and answers "yes" with } m \text{ in a } \mathbf{PA}^{>\frac{1}{2}}_{\Sigma_1 \wedge \Pi_1} \text{ game } \}.$$

Since the halting problem of $\mathcal{R}$ is Σ_1^0-complete, any winning strategy for player II is Σ_1^0-hard. □

By similar argument, we have the following reverse mathematical results. Note that our results on probabilistic automata of strict threshold can also hold for the hierarchical probabilistic automata of strict threshold. The latter is less expressive than the former.

Theorem 4.4. *The following diagram holds.*

$$\begin{array}{lcccl}
\mathbb{L}(\mathbf{PA}^{>\frac{1}{2}}_{\Sigma_2})\text{-Det} & \leftarrow & \mathsf{ATR}_0 & \rightarrow & \mathbb{L}(\mathbf{HPA}^{>\frac{1}{2}}_{\Sigma_2})\text{-Det} \\
\mathbb{L}(\mathbf{PA}^{>\frac{1}{2}}_{\Delta_2})\text{-Det} & \rightarrow & \Delta^0_2\text{-Det} & \rightarrow & \mathbb{L}(\mathbf{HPA}^{>\frac{1}{2}}_{\Delta_2})\text{-Det} \\
\mathbb{L}(\mathbf{PA}^{>\frac{1}{2}}_{\Sigma_1\wedge\Pi_1})\text{-Det} & \leftrightarrow & \mathsf{ACA}_0 & \rightarrow & \mathbb{L}(\mathbf{HPA}^{>\frac{1}{2}}_{\Sigma_1\wedge\Pi_1})\text{-Det} \\
\mathbb{L}(\mathbf{PA}^{>\frac{1}{2}}_{\Sigma_1})\text{-Det} & \leftrightarrow & \mathsf{WKL}_0 & \leftrightarrow & \mathbb{L}(\mathbf{HPA}^{>\frac{1}{2}}_{\Sigma_1})\text{-Det}
\end{array}$$

where WKL_0 *(weak König lemma) states that every infinite binary tree has an infinite path,* ACA_0 *allows the existence of all arithmetical sets, and* ATR_0 *asserts the existence of a transfinite hierarchy produced by iterating arithmetical complexity along a given well-order, all of which are popular subsystems of reverse mathematics.*

5 Future work

We studied determinacy of probabilistic ω-languages of strict threshold semantics, which is an extension of our previous results on the determinacy of pushdown ω-language in lower arithmetical hierarchy in [9]. As we illustrated in Figure 5, the determiancy of ω-languages with a Büchi or Muller have been well investigated, where the classes of ω-languages below the dotted line is determined while the ones above are not. Our next goal is to pin down the determinacy strength of ω-languages with a Büchi condition of other semantics, such as positive, almost sure and strict semantics, as indicated in the grey part of Figure 5.

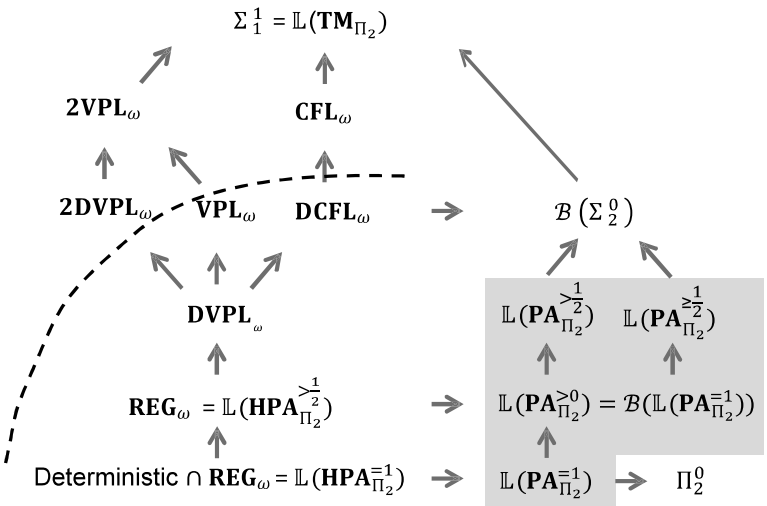


Figure 5: The inclusion relation among several classes of ω-languages with Büchi (Π_2) acceptance condition (except that $\mathbf{DCFL}_\omega$ and $\mathbf{DVPL}_\omega$ are defined with a Muller condition), where $A \rightarrow B$ means $A \subset B$. (**D**)$\mathbf{CFL}_\omega$ denotes the class of ω-languages accepted by (deterministic) pushdown automata. (2/**D**)$\mathbf{VPL}_\omega$ denotes the class of ω languages accepted by (2-stack/deterministic) visibly pushdown automata. The visibly pushdown automata is a special kind of pushdown automata in which the input alphabet is partitioned according to the operation on the stack.

Acknowledgement

The work was supported by the Natural Science Foundation of Beijing, China (Grant No. 1234038, IS23001). This work was presented at RIMS workshop: New frontiers of proof and computation held in the Research Institute for Mathematical Sciences, an International Joint Usage/Research Center located in Kyoto University.

References

[1] J.R. Büchi and L.H. Landweber, Solving sequential conditions by finite-state strategies. *Trans. Amer. Math. Soc.* **138** (1969) 295-311.

[2] Chadha, Rohit, Sistla, A. Prasad, Viswanathan, Mahesh (2009). On the expressiveness and complexity of randomization in finite state monitors. Journal of the ACM, 56(5), 1-44.

[3] Baier Christel, Grösser Marcus, Bertrand Nathalie, 2012. Probabilistic ω-automata. Journal of the ACM (JACM), 59(1), 1-52.

[4] K. Chatterjee, T.A. Henzinger, 2010. Probabilistic Automata on Infinite Words: Decidability and Undecidability Results, in: Bouajjani, A., Chin, W.-N. (Eds.), Automated Technology for Verification and Analysis, Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, pp. 1-16.

[5] Chadha, R., Viswanathan, M., 2011. Power of Randomization in Automata on Infinite Strings. *Log. Methods Comput. Sci* , 7(22:3), 1-31. Logical Methods in Computer Science

[6] Chadha, R., Sistla, A.P., Viswanathan, M.: Probabilistic Büchi automata with non-extremal acceptance thresholds. In: International Workshop on Verification, Model Checking, and Abstract Interpretation. pp. 103-117. Springer (2011)

[7] Phan Dinh Diêu (1971). On the Languages Representable by Finite Probabilistic Automata. Zeitschrift fur mathematische Logik und Grundlagen der Mathematik 17 (1):427-442.

[8] O. Finkel, The determinacy of context-free games. *J. Symb. Log.* **78** (2013) 1115-1134.

[9] W. Li, K. Tanaka, The determinacy of context-free games. *RAIRO Inform. Théor. Appl.* **51** (1) , 2017, 29-50.

[10] C. Löding, A. Pirogov, Ambiguity, Weakness, and Regularity in Probabilistic Büchi Automata. FOSSACS 2020, LNCS 12077, pp. 522-541, 2020.

[11] M.L. Minsky, Recursive unsolvability of Post's problem of tag and other topics in theory of Turing machines. *Ann. of Math.* (1961) 437-455.

[12] M.L. Minsky, *Computation: finite and infinite machines.* Prentice-Hall, Inc. (1967).

[13] T. Nemoto, M.Y. Ould MedSalem and K. Tanaka, Infinite games in the Cantor space and subsystems of second order arithmetic. *MLQ Math. Log. Q.* **53** (2007) 226-236.

[14] A. Paz Introduction to probabilistic automata, Computer Science and Applied Mathematics, 1971.

[15] M.O. Rabin, Probabilistic automata, nformation and Control. 6 (3): 230-245, 1963.

[16] L. Staiger, ω-languages, in Chapter 6 of the *Handbook of Formal Languages*, vol. 3, edited by G. Rozenberg and A. Salomaa, Springer-Verlag, Berlin (1997) 339-387.

[17] I. Walukiewicz, Pushdown processes: Games and model-checking, in *International Conference on Computer Aided Verification* Springer Berlin Heidelberg (1996) 62-74. *Inform. and Comput.* **164** (2001) 234-263.